INFORMATION AND COMPUTATION 103, 195–203 (1993)

Limitations of the Program Memory and the Expressive Power of Dynamic Logics*

I. KH. MUSIKAEV AND M. A. TAITSLIN

Department of Applied Mathematics and Cybernetics, Tver State University, Tver 170013, Russia

For a dynamic logic L we study dynamic logics L_n for which programs allowed in formulas cannot use more than n variables. We prove that there exists a structure A of a finite signature such that for a wide class of dynamic logics L and for every natural number n the logic L_{n+1} is more expressive over A than L_n. This result is based on a construction of some canonical form for the formulas of L_n over a free one-generated groupoid.

INTRODUCTION

The influence of program memory limitations on the expressive power of programming logics was studied by many authors. Paterson and Hewitt (1970), and independently Friedman (1971), first showed that programs using an unbounded memory have greater computational power than programs using a bounded amount of memory. Later, a similar question was considered for dynamic logics. For the various program constructions giving an unbounded memory, Harel (1979), Meyer and Winklmann (1982), Kfoury (1983), Tiuryn (1984), and others have proved that dynamic logics with unbounded program memory are more expressive than dynamic logics with bounded program memory.

Unbounded program memory allows a potentially infinite amount of memory to be used during computation. From a practical point of view it may be more natural to examine the effect of adding of a finite amount of memory. More formally, given a dynamic logic L with bounded program memory and a natural number n, we denote by L_n the dynamic logic whose formulas consist of those and only those formulas of L which use no more than n variables in programs. Then the question is whether $L_{n+1} > L_n$ holds for every n? As usual, if $\mathbf{C}$ is a class of structures of a signature σ then a logic L is more expressive than a logic L' over $\mathbf{C}$ ($L >_{\mathbf{C}} L'$) if for

* A short abstract of this paper has been presented in Musikaev and Taitslin (1986).

0890-5401/93 $5.00

every formula ϕ of L' of a signature σ there exists a formula of L, equivalent to ϕ over all structures from $\mathbf{C}$, but not vice versa. If the subscript $\mathbf{C}$ is omitted then the class of all structures is considered.

In such a formulation the solution of the problem is simple.

Let B_n be the union of an infinite "ω-chain" of copies of the full binary tree of depth n, the root of each copy attached at the leftmost leaf of the next copy. Let $\mathbf{B}_n$ be a structure $(B_n, \cdot, ')$, where

(i) $(t_1 \cdot t_2)$ is the tree with left child t_1 and right child t_2 for $(t_1 \cdot t_2) \in B_n$ and t_1 otherwise,

(ii) t' is the next leaf for all leaves of each binary tree except the rightmost leaves and t otherwise.

It is not hard to exhibit a formula of L_{n+1} which defines B_n uniquely up to an isomorphism. On the other hand, the standard reasoning about the register complexity of terms allows it to be shown that there are only a finite number of terms whose register complexity does not exceed n. Therefore, each formula of L_n is equivalent to a first-order formula over $\mathbf{B}_n$ and every structure has been elementarily equivalent to $\mathbf{B}_n$. But there is an uncountable structure elementarily equivalent to $\mathbf{B}_n$. Therefore, $L_{n+1} > L_n$.

But this solution of the problem is not a good one, since we actually have proved that, for a given n, there is a structure where $n+1$ variables are "better" than n variables. But $n+2, n+3, \ldots$ may not be better than $n+1$ variables over this structure. For the full solution of the problem it is necessary to prove that there exists a single structure $\mathbf{A}$ over a finite signature such that, for every n, the logic L_{n+1} is more expressive over $\mathbf{A}$ than L_n.

We prove that L_{n+1} is more expressive than L_n over the free one-generated groupoid $\mathbf{G}$ for a wide class of dynamic logics L with bounded memory (the free one-generated groupoid is considered only for simplicity, since any free finitely generated algebra of a nonunary signature is an appropriate structure).

The proof of this result is based on a careful analysis of the finite register (pebble) complexity formulas of logic with infinite conjunctions over $\mathbf{G}$, which is given in Theorem 1 of Musikaev and Taitslin (1989). This theorem generalizes a well-known result of Maltsev (1961) that over every locally free algebra each formula of first-order logic is equivalent to a certain canonical form.

We prove an analogous fact for any bounded register complexity formula of the logic $L_{\infty\omega}$ and any free finitely generated algebra of nonunary signature (though every dynamic logic with bounded memory is reducible to $L_{\omega_1\omega}$, the reference to $L_{\infty\omega}$ is necessary in the induction hypothesis used in the proof, since canonical forms of formulas in $L_{\omega_1\omega}$ are not necessary in $L_{\omega_1\omega}$).

Using this fact we can also prove that dynamic logics with unbounded program memory are more expressive than dynamic logics with bounded program memory.

One of the consequences of the result about the comparison of logics L_n and L_{n+1} is as follows.

In a number of papers (for example, Harel and Chandra (1982), Stolboushkin and Taitslin (1986)) dynamic logics are considered as query languages for data bases. In this context the following question naturally arises: Given a dynamic logic L and a structure A is there a finite set of formulas (read queries) such that for any formula (query) of L one can construct an equivalent formula (query) over A from the formulas (queries) of this set by means of disjunction, conjunction, negation, and quantification (read by means of unity, intersection, complement, and projection)? Stolboushkin (1986) showed that this question has a positive answer for any structure of unary function finite signature and deterministic dynamic logic. We prove that it has a negative answer for a wide class of logics and the one-generated free groupoid.

1. Preliminaries

In this section we give definitions and some basic facts concerning the notion of register complexity of terms. If necessary the reader can find the proof of these facts in Musikaev and Taitslin (1989). We also formulate the theorem about canonical form of formulas of $L_{\infty\omega}$ over a free finitely generated algebra of nonunary signature (Theorem 1) which is used to prove the main results of the paper. The proof of this theorem is rather complicated and exceeds the limits of this paper. The full proof is in Musikaev and Taitslin (1989).

Let $x, y, z_1, ..., z_n$ be variables, some of which can coincide, let f be an n-ary functional symbol, and let c be a constant symbol. Expressions of the form

$$x := y, \qquad x := c, \qquad x := f(z_1, ..., z_n)$$

are called assignments.

A signature (similarity type) is a finite sequence of function, relation, and constant symbols, each nonconstant symbol with a nonnegative arity. To every sequence α of assignments using variables only from $(x_1, ..., x_n)$ and constant and function symbols only from a signature σ, we assign a function $v(\alpha)$ from $(x_1, ..., x_n)$ to the set of all terms of signature σ. This function is defined by the induction on the length s of the sequence α.

If α is empty ($s = 0$) then $v(\alpha)(x_i) = x_i$. If $\alpha = \alpha' a_s$ then one of the three following cases takes place.

Case 1. If α_s is $x_i := x_j$, then $v(\alpha)(y) = v(\alpha')(y)$ for $y \in (x_1, ..., x_n) - (x_i)$ and $v(\alpha)(x_i) = v(\alpha')(x_j)$.

Case 2. If α_s is $x_i := c$, where c is a constant symbol of σ then $v(\alpha)(y) = v(\alpha')(y)$, for $y \in \{x_1, ..., x_n\} - \{x_i\}$ and $v(\alpha)(x_i) = c$.

Case 3. If α_s is $x_i := f(x_{j_1}, ..., x_{j_m})$, where f is an m-ary functional symbol from σ, and $j_1, ..., j_m \in (1, ..., n)$, then $v(\alpha)(y) = v(\alpha')(y)$, for $y \in \{x_1, ..., x_n\} - \{x_i\}$, and

$$v(\alpha)(x_i) = f(v(\alpha')(x_{j_1}), ..., v(\alpha')(x_{j_m})).$$

A sequence α of assignments computes a term t in a variable y iff t is $v(\alpha)(y)$.

The register complexity of a term t (denoted $\mathrm{rc}(t)$) is the least number n such that t is computed by a sequence of assignments using at most n variables.

The notion of a formula of signature σ of logic $L_{\infty\omega}(n)$ is defined as follows. If F is an atomic formula of signature σ and the register complexity of every term occurring in F is not more than n then F is a formula of signature σ of $L_{\infty\omega}(n)$. In this case $d(F) = 0$. If Φ is a set of formulas of signature σ and there exists such a number m that $d(F) < m$, for every F from Φ, then

$$\bigvee_{F \in \Phi} F, \qquad \bigwedge_{F \in \Phi} F$$

are formulas of signature σ of $L_{\infty\omega}(n)$. In this case $d(\bigvee_{F\in\Phi} F) = d(\bigwedge_{F\in\Phi} F) = (\max_{F\in\Phi} d(F)) + 1$. If x is a variable and F is a formula of signature σ of $L_{\infty\omega}(n)$, then $\neg F$, $(\forall x) F$, $(\exists x) F$ are formulas of signature σ of $L_{\infty\omega}(n)$. In this case $d(\neg F) = d((\forall x) F) = d((\exists x) F) = d(F) + 1$.

For a term t we write $t(x_1, ..., x_s)$ to indicate that all variables occurring in t are among $\{x_1, ..., x_s\}$. If $t, t_1, ..., t_s$ are terms and $x_1, ..., x_s$ are variables then $t(t_1/x_1, ..., t_s/x_s)$ denotes the term obtained from t by substituting t_i for every occurrence of x_i in t, for every $1 \leqslant i \leqslant s$.

The composition degree of a term t with respect to n, denoted by $\mathrm{cd}(t, n)$ is the least number m such that t is a composition of m terms, whose register complexity is less or equal to n. In more detail, if $\mathrm{rc}(t) \leqslant n$ then $\mathrm{cd}(t, n) = 1$. Otherwise, $\mathrm{cd}(t, n)$ is the least number m such that t is $t_1(t_2/x)$, for some simpler terms t_1, t_2, and $m = \mathrm{cd}(t_1, n) + \mathrm{cd}(t_2, n)$.

Let σ_0 be the signature consisting of the constant symbol a and the binary function symbol $\cdot$. By T we denote the set of all terms of signature σ_0. We use the infix notation of terms; i.e., by a term of signature σ_0 we mean either a variable or constant c or an expression of the form $(t_1 \cdot t_2)$, where t_1 and t_2 are terms.

By $f(i)$ we denote the term defined inductively as follows: $f(0) = x_1$, $f(i+1) = (f(i) \cdot (f(i)(x_{s+1}/x_1, ..., x_{2s}/x_s)))$, where $s = 2^i$. If one represents terms by finite binary trees (see Tiuryn, 1984, 2.3.) then $f(i)$ is the full binary tree of depth i, whose different nodes are marked by different variables. If every leaf of the full binary tree of depth i is marked by a then it represents the term denoted by $\underline{i}$. Thus, $\underline{i} = f(i)(a/x_1, ..., a/x_s)$, where $s = 2^i$. The register complexity of $\underline{i}$ is equal to 1.

Terms t_1 and t_2 will be called independent iff they are different. Let $s = 2r$, $r = 2^i$. Terms $t_1, ..., t_s$ will be called independent iff $t_1, ..., t_r$ are independent, $t_{r+1}, ..., t_s$ are independent, and:

(a) for every subterm t' of term $f(i)$ the term $t'(t_1/x_1, ..., t_r/x_r)$ differs from $t_{r+1}, ..., t_s$;

(b) for every subterm t' of the term $f(i)$ the term $t'(t_{r+1}/x_1, ..., t_s/x_r)$ differs from $t_1, ..., t_r$.

LEMMA 1. *Let $s = 2^i$ and $t_1, ..., t_s$ be independent terms of signature σ_0, $t = f(i)(t_1/x_1, ..., t_s/x_s)$. Then* $\mathrm{rc}(t) \geqslant i + 1$.

This result is implicitly contained in Friedman (1971) and Paterson and Hewitt (1970). Proposition 2.10 in Tiuryn (1984) differs from Lemma 1 only by substitution of "pairwise different" for "independent." The following example shows that this is not correct.

It is clear that $f(2) = ((x_1 \cdot x_2) \cdot (x_3 \cdot x_4))$. Let $t_1 = a$, $t_2 = x$, $t_3 = (a \cdot x)$, $t_4 = ((a \cdot x) \cdot (a \cdot x))$. Terms t_1, t_2, t_3, t_4 are pairwise different. But $\mathrm{rc}(t) = 2$ for $t = f(2)(t_1/x_1\ t_2/x_2, t_3/x_3, t_4/x_4) = ((a \cdot x) \cdot ((a \cdot x) \cdot ((a \cdot x) \cdot (a \cdot x))))$.

Indeed,

$$y := a,\ y := (y \cdot x),\ x := (y \cdot y),\ x := (y \cdot x),\ x := (y \cdot x)$$

computes t in x.

By $f(n, i)$ we denote the term $f(n)(\underline{s \cdot i + 1}/x_2, ..., \underline{s \cdot (i+1)}/x_{s+1})$, where $s = 2^n - 1$, and i is a natural number. Let $g(m, n, 0) = f(n, m)$, $g(m, n, i+1) = f(n, m+i+1)(g(m, n, i)/x_1)$, and $\underline{(i, n)} = g(0, n, i)(a/x_1)$.

LEMMA 2. *For all natural numbers i, n, m,* $\mathrm{rc}(g(m, n, i)) = \mathrm{rc}(\underline{(i, n)}) = n + 1$.

In what follows we use the notation $\eta(i) = 2^{i+1} - 1$.

LEMMA 3. *Either let t be the constant a or let t contain a variable and not*

contain subterms of $g(m, n, i)$, *whose register complexity is more than* 1. *Then*

$$\mathrm{cd}(g(m, n, \eta(i))(t_1/x_1), n) > i + 1.$$

In particular, $\mathrm{cd}((\underline{\eta(i), m}), m) > i + 1$ *for all natural numbers* i *and* m.

Let σ be a signature, let I be a finite set of variables, and let m, n, q be natural numbers. A formula of the form

$$(\exists y_1)\cdots(\exists y_l)\left(\left(\bigwedge_{x\in I_1} x = t_x\right)\wedge\left(\bigwedge_{x\in I_2}\left(\bigwedge_{\alpha\in R_x} x \neq t_\alpha^x\right)\right)\right.$$
$$\left.\wedge\left(\bigwedge_{j=1}^{l}\left(\bigwedge_{\alpha\in Q(j)} y_j \neq t_\alpha^j\right)\right)\right)$$

will be called an (I, m, n, q)-standard formula if the following conditions hold:

0. $l \leqslant m$;
1. $I = I_1 \cup I_2$, $In(y_1, ..., y_l) = \varnothing$, $I_1 n I_2 = \varnothing$;
2. $\mathrm{cd}(t_x, n) \leqslant q$, for every $x \in I_1$;
3. $\mathrm{cd}(t_\alpha^x, n) \leqslant q$, for every $x \in I_2$ and $\alpha \in R_x$;
4. $\mathrm{cd}(t_\alpha^j, n) \leqslant q$ for every $j \in \{1,..., l\}$ and $\alpha \in Q(j)$;
5. every variable occurring in any term t_x, t_α^x, t_α^x occurs in $\{y_1, ..., y_l\} \cup I_2$;
6. every variable y_i $(1 \leqslant i \leqslant l)$ occurs in some term t_x $(x \in I_2)$.

A disjunction (possibly infinite) of (I, M, n, q)-standard formulas will be called an (I, M, n, q)-canonical formula.

THEOREM 1. *For every formula* θ *of* $L_{\infty\omega}(n)$ *with the set* I *of free variables, and for every free finitely generated algebra* $\mathbf{A}$ *of any nonunary signature, there are such natural numbers* m *and* q *that* θ *is equivalent to some* (I, m, n, q)*-canonical formula of* $L_{\infty\omega}$ *over* A.

This theorem has been proved in Musikaev and Taitslin (1989). Conjunctions and disjunctions in the canonical formula from Theorem 1 may be uncountable even if formula θ belongs to $L_{\omega_1\omega}$. Therefore we can not replace $L_{\infty\omega}(n)$ by $L_{\omega_1\omega}(n)$.

2. MAIN RESULTS

The reader is assumed to be familiar with basic notions concerning dynamic logics. Here we give only a brief definition of the syntax and

semantic of dynamic logics and refer the reader to Harel (1979) and Stolboushkin and Taitslin (1986) for detailed definitions.

Let K be some class of programs. The dynamic logic of the class K, denoted by DL_K, is defined as follows.

1. Every first-order formula is a formula of DL_K.

2. If $P \in K$ is a program and F is a formula of DL_K, then $\langle P \rangle F$ is a formula of DL_K.

3. If F', F'' are formulas of DL_K and x is a variable then $(F' \vee F'')$, $(F' \wedge F'')$, $\neg F'$, $(\exists x) F'$, $(\forall x) F'$ are formulas of DL_K.

The meaning of a formula $\langle P \rangle F$ for a given valuation v is that there is a computation of the program P which starts in v and terminates with a valuation satisfying F.

The class of all regular programs is denoted by RG; the class of all recursively enumerable programs (regular programs with counters) is denoted by RE.

By a class of programs with bounded memory we understand every class of programs reducible to RE. If a dynamic logic L is expressive equally to some dynamic logic DL_K, where K is a class of programs with bounded memory, then L is a dynamic logic with bounded memory.

We say that a program uses n variables if it is equivalent to an RE-program which actually has n variables. If K is a class of programs with bounded memory then the class of all programs from K which use no more than n variables will be denoted by K_n.

The one-generated free groupoid $\mathbf{G}$ is the structure of signature σ_0 with the domain T_0 of all variable-free terms of signature σ_0 and the natural interpretation of constant a and function $\cdot$.

Let $H_n = \{\underline{(i, n)} \mid i = 0, 1, \ldots\}$.

LEMMA 4. *For every number n, there is no formula of $L_{\infty\omega}(n)$ which defines the set H_n in $\mathbf{G}$ (i.e., is satisfied by those and only those elements of G which belong to H_n).*

Proof. Let such a formula $\theta(x)$ exist. The groupoid $\mathbf{G}$ is a free finitely generated algebra of nonunary signature, so $\theta(x)$ is equivalent over $\mathbf{G}$ to some $(\{x\}, m, n, q)$-canonical formula $\bigvee_{\alpha \in \Gamma} \theta_\alpha$, where θ_α is an $(\{x\}, m, n, q)$-standard formula. Every θ_α must be a quantifier-free formula. Indeed, let θ_α be $(\exists y_1) \cdots (\exists y_l)\, \theta''_\alpha$, where $l \geqslant 1$ and θ''_α is

$$x = t_\alpha \wedge \left(\left(\bigwedge_{\beta \in \Delta_1} y_1 \neq t_1^\beta \right) \wedge \cdots \wedge \left(\bigwedge_{\beta \in \Delta_l} y_l \neq t_l^\beta \right) \right).$$

There is such an element, say $b \in G$, that θ_α is satisfied in $\mathbf{G}$ for $x = b$. Then

there exist such elements $b_1, ..., b_l$ in G that for $x = b$, $y_1 = b_1, ..., y_l = b_l$ the formula θ''_α is satisfied in **G**. Now we choose such b'_l that the register complexity of b'_l is greater than $(q \cdot n + \sum_{j=1}^{l-1} \mathrm{rc}(b_j) + 2)$. It is clear that the formula θ_α is satisfied on the element $t(b_1/y_1, ..., b_{l-1}/y_{l-1}, b'_l/y_l)$ too, but this element does not belong to H_n.

Therefore θ_α is $x = t_\alpha$ and θ is $\bigvee_{\alpha \in \Gamma} x = t_\alpha$, where $\mathrm{cd}(t_\alpha, n) \leqslant q$, for every $\alpha \in \Gamma$. But it follows from Lemma 3 that θ is not satisfied by $\underline{(\eta(2q), n)}$ which belongs to H_n. This proves the lemma. ∎

THEOREM 2. *Let K be a class of programs with bounded memory. If* RG *is reducible to K then for every natural number i, the logic* $\mathrm{DL}_{K_{i+1}}$ *is more expressive over* **G** *than* DL_{K_i}.

Proof. First, we construct the formula of RG_{i+1} which define the set H_i in G. Let $h(i)$ be the term $f(i)(g(2)/x_2, ..., g(s)/x_s)$, where $s = 2^i$, $g(2) = x_2$, $g(i+1) = (g(i) \cdot g(i))$. It is obvious that $h(i)(\underline{(2^n - 1) \cdot i + 1/x_2}) = f(n, i)$. Taking into account Lemma 2 it is easy to construct a sequence α_i of assignments which contains $i+1$ variables, and computes $g(2^i + 1)$ in x_2 and $h(i)$ in x_1. An example of such a sequence for $i = 2$ will be

$$x_1 := (x_1 \cdot x_2), \quad x_3 := (x_2 \cdot x_2), \quad x_2 := (x_3 \cdot x_3),$$
$$x_3 := (x_3 \cdot x_2), \quad x_1 := (x_1 \cdot x_3), \quad x_2 := (x_2 \cdot x_2).$$

The formula $F(x) = \langle x_1 := a;\ x_2 := (x_1 \cdot x_1);\ \alpha_i^* \rangle\, x_1 = x$ defines H_i in G.

Every formula of DL_{K_i} is equivalent to some formula of $L_{\infty\omega}(i)$. (The proof of this proposition immediately follows from the definition of the class K_n. The reader can find a similar proof in Lemma 3 of Meyer and Parikh 1981.) This is why, using Lemma 4, we obtain the result that there is no formula of DL_{K_i} which takes out H_i in **G**. This proves the theorem. ∎

We say that a dynamic logic DL_{K_1} is essentially more expressive over a structure **A** than a dynamic logic DL_{K_2} iff for every finite set $K \subseteq K_1$ of programs, DL_{K_1} is more expressive over **A** than $\mathrm{DL}_{K \cup K_2}$.

If a dynamic logic DL_K used as a query language for a data base (see, for example, Harel and Chandra, 1982; Stolboushkin and Taitslin, 1986) is not essentially more expressive than first-order logic over some structure **A** (we can consider first-order logic as $\mathrm{DL}_\varnothing$) then it means that, for the data base **A** there is a finite set K of queries from which one can construct every query of this language by the main operations of relational data bases (unity, intersection, complement, and projection).

is reducible to K then DL_K is essentially more expressive over **G** *than classical first-order logic.*

Proof. Let K' be an arbitrary finite subset of K. There exists n such that $K' \leqslant K_n$. But $DL_{K_{n+1}}$ is more expressive over **G** than DL_{K_n}, and consequently $DL_{K_{n+1}}$ is more expressive over **G** than $DL_{K'}$. Therefore DL_K is more expressive than $DL_{K'}$.

ACKNOWLEDGMENTS

The authors express their gratitude to J. Tiuryn and P. Urzyczyn for helpful comments.

REFERENCES

FRIEDMAN, H. (1971), Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory, *in* "Logic Colloqium '69" (Gandy and Yates, Eds.) pp. 361–390, North-Holland, Amsterdam.

HAREL, D. (1979), "First-Order Dynamic Logic," Springer, Berlin.

HAREL, D. AND CHANDRA, A. (1982), Structure and complexity of relational queries, preprint, Weizmann Institute of Science, CS82-05.

KFOURY, A. J. (1983), Definability by programs in first-order structures, *Theoret. Comput. Sci.* **25**, 1–66.

MALTSEV, A. I. (1961), On elementary theories of local free universal algebras, *Dokl. Akad. Nauk SSSR* **138**, 1009–1012. [Russian]

MEYER, A. R. AND PARIKH, R. (1981), Definability by dynamic logic, *J. Comput. System Sci.* **23**, 279–298.

MEYER, A. R. AND WINKLMANN, K. (1982), Expressing program looping in regular dynamic logic, *Theoret. Comput. Sci.* **18**, 301–323.

MUSIKAEV I. KH. AND TAITSLIN, M. A. (1989), On dynamic theories of free algebras, *Mat. Sb.* **180** (No. 3), 307–321. [Russian]

MUSIKAEV, I. KH. AND TAITSLIN, M. A. (1986), Dynamic theories of free algebras, *in* "4th All-Union Conference on Application of Methods of Mathematical Logic, Tallinn, May, 1986," pp. 123–125.

PATERSON, M. S. AND HEWITT, C. E. (1970), Comparative schematology, *in* "Record of Project MAC Conference on Concurrent System and Parallel Computation, ACM, New York, December 1970," pp. 119–128.

STOLBOUSHKIN, A. P. (1986), Dynamic expansions of unoids is almost elementary trivial, *in* "4th All-Union Conference on Applications of Methods of Mathematical Logic, Tallinn, May, 1986."

STOLBOUSHKIN, A. P. AND TAITSLIN, M. A. (1986), Dynamic logics, *in* "Cybernetics and Computers" (V. A. Melnikov, Ed.), pp. 180–230, Nauka, Moscow. [Russian]

TIURYN, J. (1984), Unbounded program memory adds to the expressive power of first-order programming logic, *Inform. and Control* **60**, 12–35.